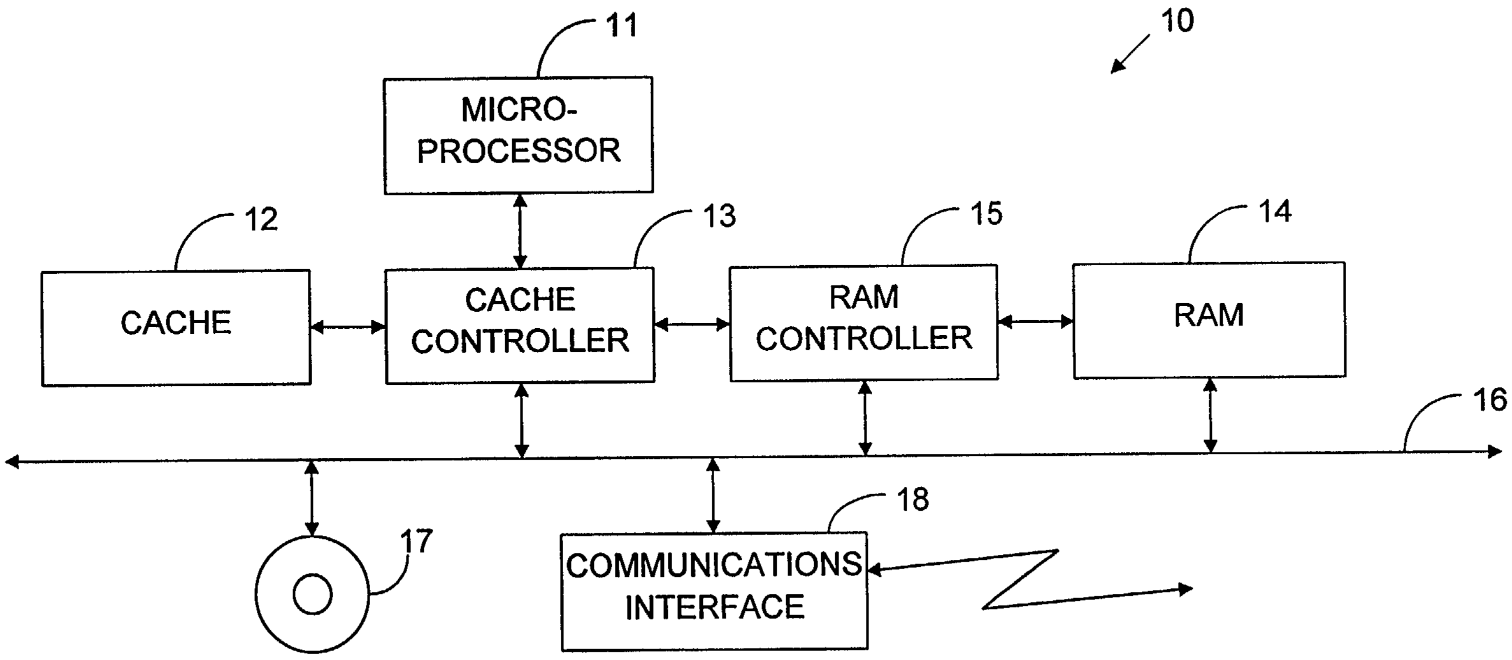

10
11
MICRO-
PROCESSOR
12
CACHE
13
CACHE
CONTROLLER
15
RAM
CONTROLLER
14
RAM
16
17
18
COMMUNICATIONS
INTERFACE

FIG. 1

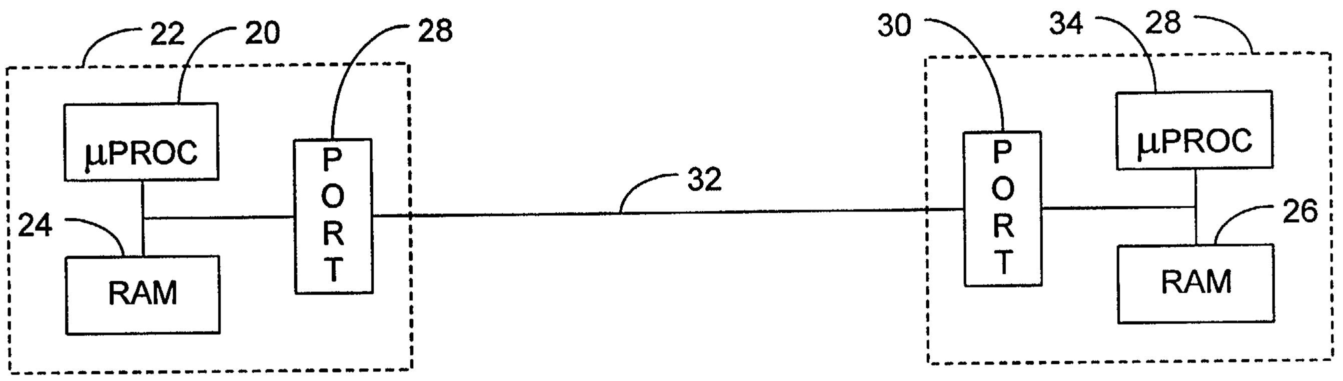

22
20
28
μPROC
RAM
24
P O R T
32
30
34
28
μPROC
P O R T
26
RAM

FIG. 2

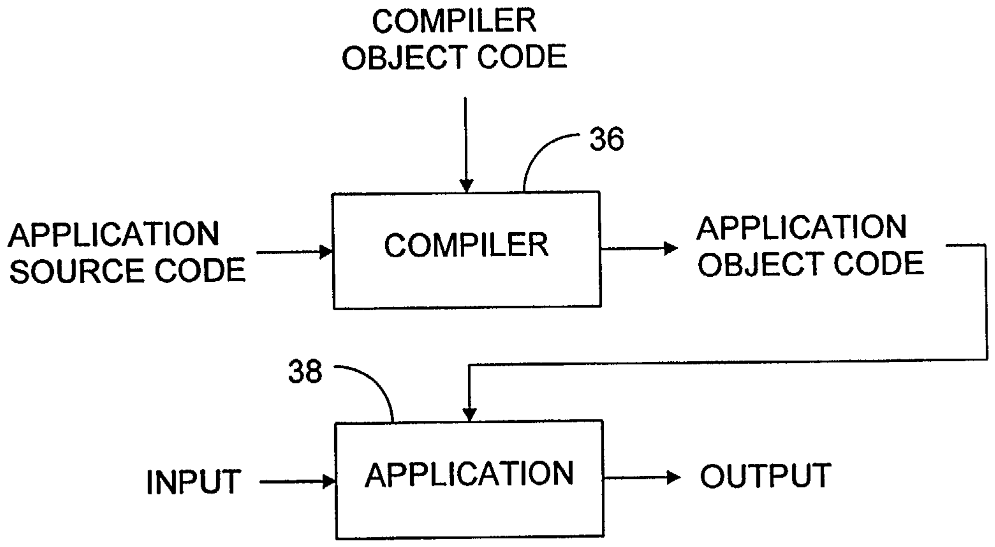

COMPILER OBJECT CODE
36
APPLICATION SOURCE CODE
COMPILER
APPLICATION OBJECT CODE
38
INPUT
APPLICATION
OUTPUT

FIG. 3

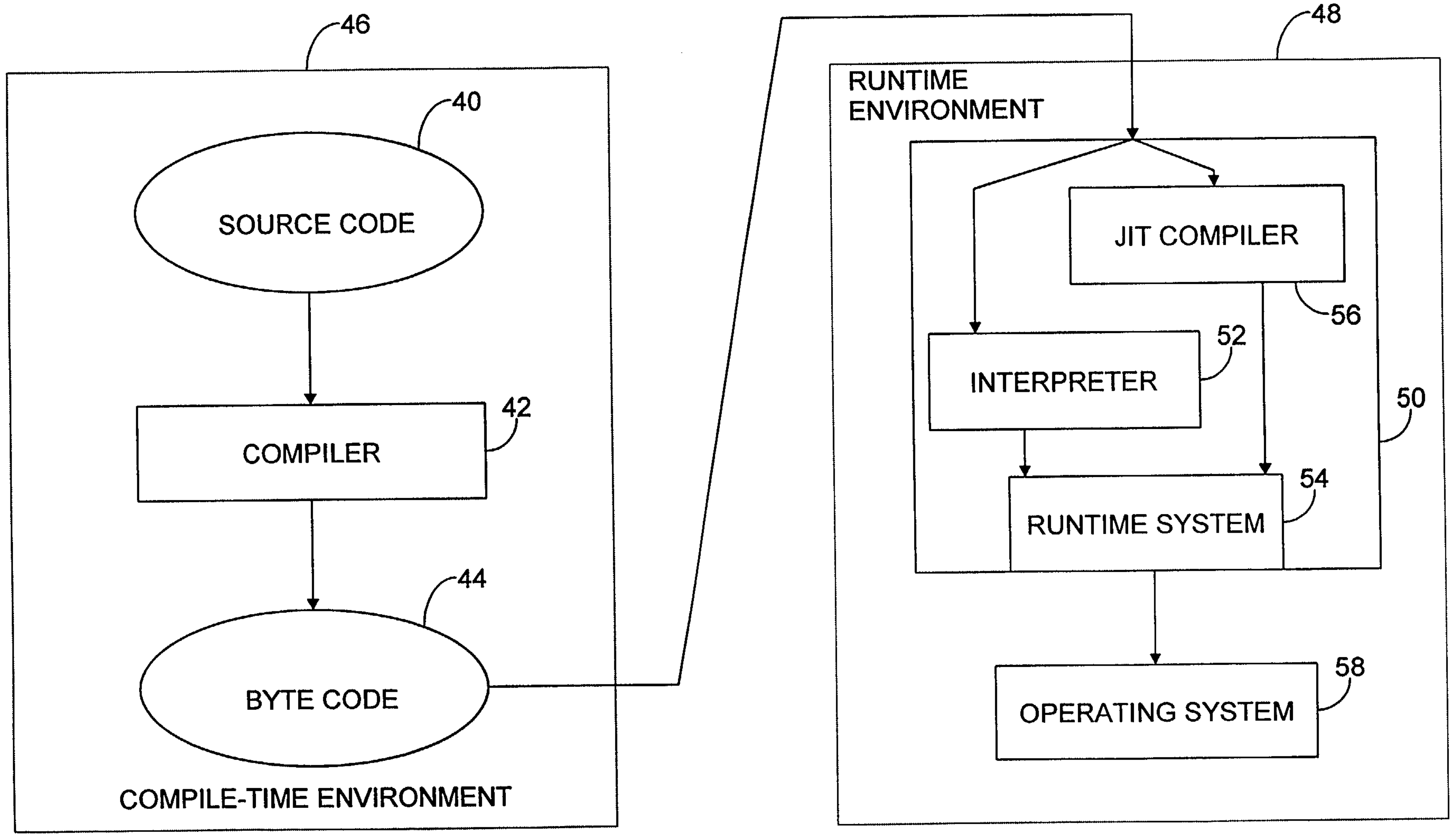
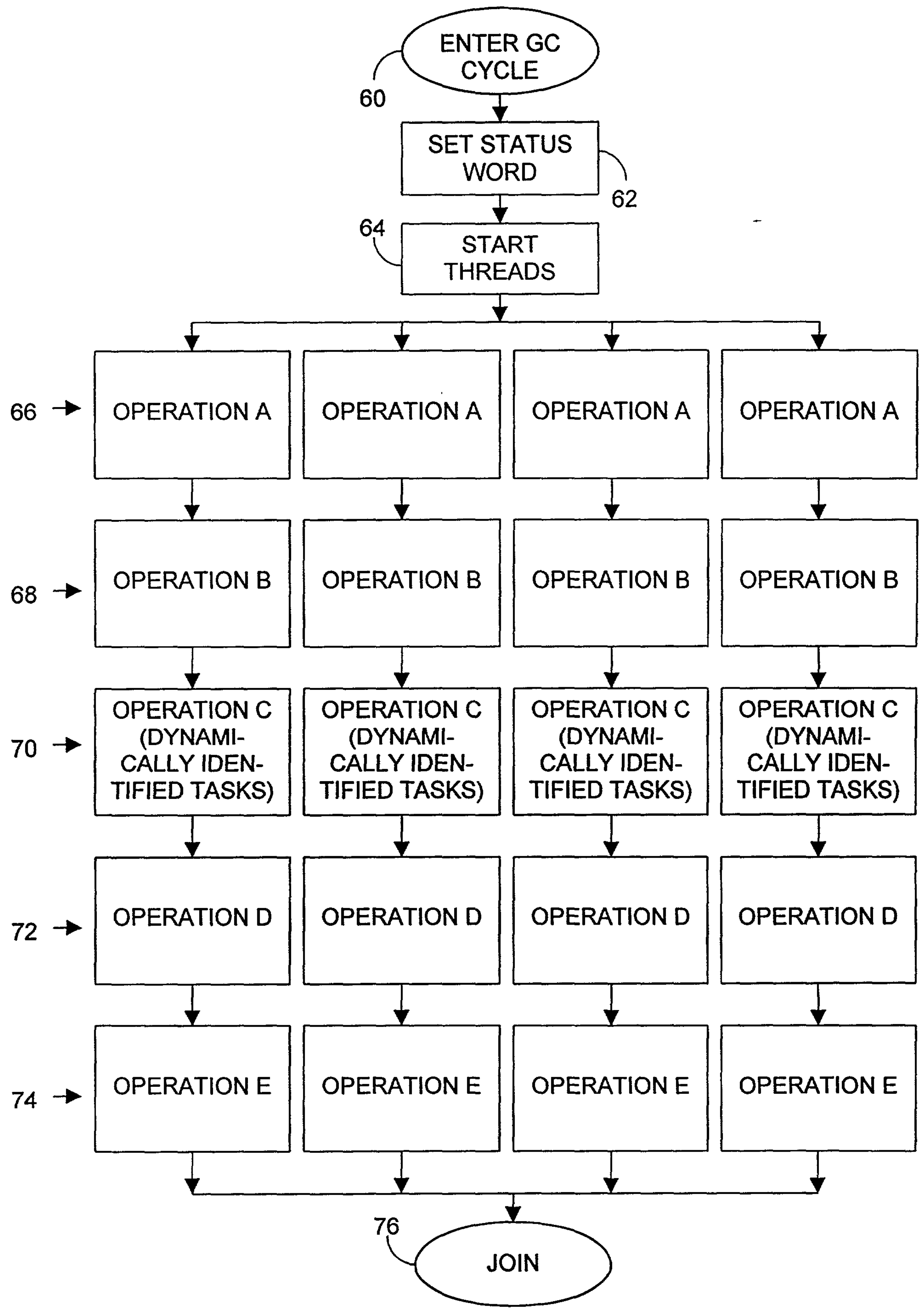

46
40
SOURCE CODE
COMPILER
42
44
BYTE CODE
COMPILE-TIME ENVIRONMENT
48
RUNTIME ENVIRONMENT
JIT COMPILER
56
INTERPRETER
52
50
RUNTIME SYSTEM
54
OPERATING SYSTEM
58

FIG. 4



ENTER GC CYCLE
60
SET STATUS WORD
62
64
START THREADS
66
OPERATION A
OPERATION A
OPERATION A
OPERATION A
68
OPERATION B
OPERATION B
OPERATION B
OPERATION B
70
OPERATION C (DYNAMI-CALLY IDEN-TIFIED TASKS)
OPERATION C (DYNAMI-CALLY IDEN-TIFIED TASKS)
OPERATION C (DYNAMI-CALLY IDEN-TIFIED TASKS)
OPERATION C (DYNAMI-CALLY IDEN-TIFIED TASKS)
72
OPERATION D
OPERATION D
OPERATION D
OPERATION D
74
OPERATION E
OPERATION E
OPERATION E
OPERATION E
76
JOIN

FIG. 5

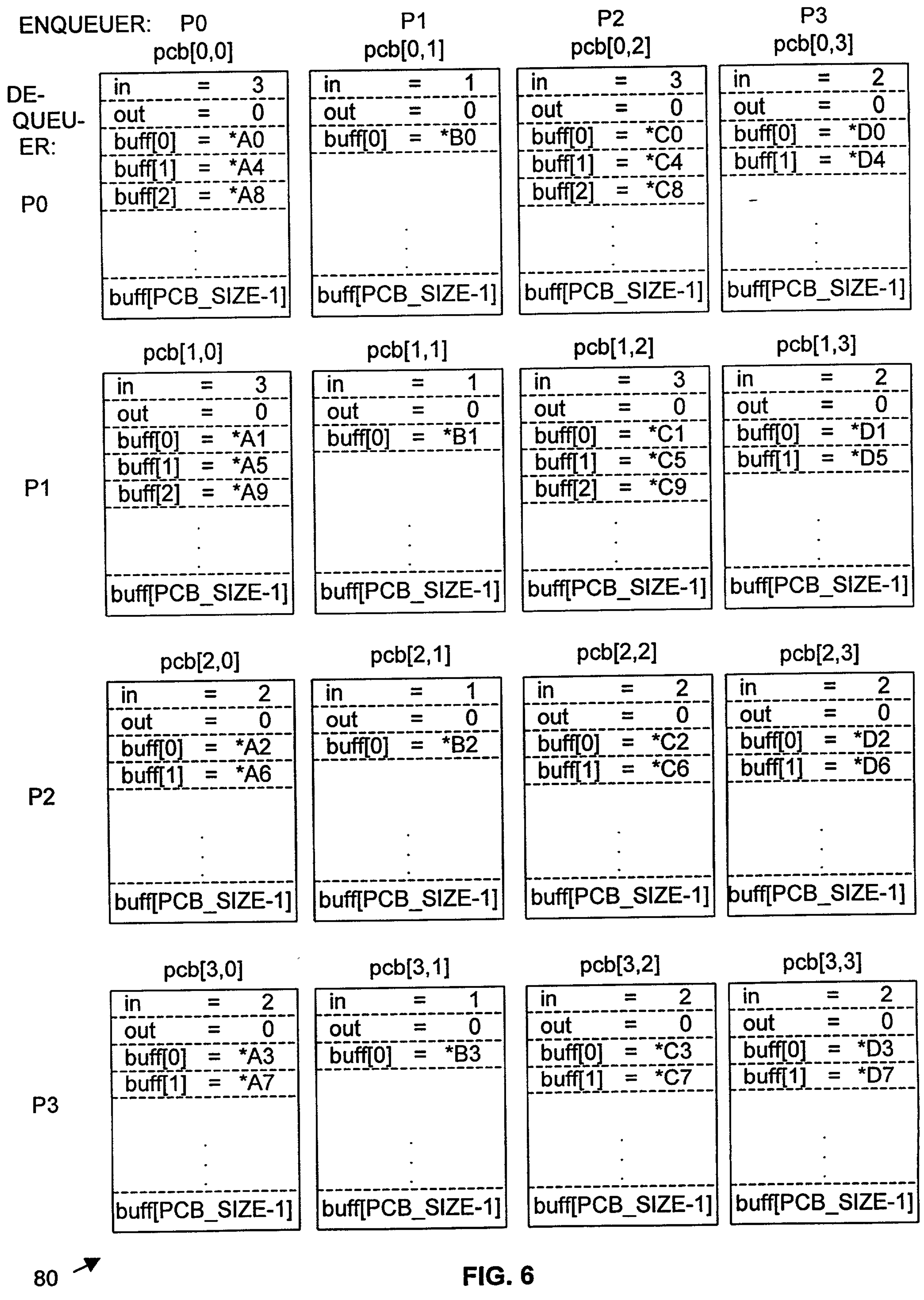

ENQUEUER:
P0
P1
P2
P3
DE-QUEUE-ER:
P0
P1
P2
P3
pcb[0,0]
in = 3
out = 0
buff[0] = *A0
buff[1] = *A4
buff[2] = *A8
buff[PCB_SIZE-1]
pcb[0,1]
in = 1
out = 0
buff[0] = *B0
buff[PCB_SIZE-1]
pcb[0,2]
in = 3
out = 0
buff[0] = *C0
buff[1] = *C4
buff[2] = *C8
buff[PCB_SIZE-1]
pcb[0,3]
in = 2
out = 0
buff[0] = *D0
buff[1] = *D4
buff[PCB_SIZE-1]
pcb[1,0]
in = 3
out = 0
buff[0] = *A1
buff[1] = *A5
buff[2] = *A9
buff[PCB_SIZE-1]
pcb[1,1]
in = 1
out = 0
buff[0] = *B1
buff[PCB_SIZE-1]
pcb[1,2]
in = 3
out = 0
buff[0] = *C1
buff[1] = *C5
buff[2] = *C9
buff[PCB_SIZE-1]
pcb[1,3]
in = 2
out = 0
buff[0] = *D1
buff[1] = *D5
buff[PCB_SIZE-1]
pcb[2,0]
in = 2
out = 0
buff[0] = *A2
buff[1] = *A6
buff[PCB_SIZE-1]
pcb[2,1]
in = 1
out = 0
buff[0] = *B2
buff[PCB_SIZE-1]
pcb[2,2]
in = 2
out = 0
buff[0] = *C2
buff[1] = *C6
buff[PCB_SIZE-1]
pcb[2,3]
in = 2
out = 0
buff[0] = *D2
buff[1] = *D6
buff[PCB_SIZE-1]
pcb[3,0]
in = 2
out = 0
buff[0] = *A3
buff[1] = *A7
buff[PCB_SIZE-1]
pcb[3,1]
in = 1
out = 0
buff[0] = *B3
buff[PCB_SIZE-1]
pcb[3,2]
in = 2
out = 0
buff[0] = *C3
buff[1] = *C7
buff[PCB_SIZE-1]
pcb[3,3]
in = 2
out = 0
buff[0] = *D3
buff[1] = *D7
buff[PCB_SIZE-1]
80

FIG. 6

```
static void enque(ParallelThread *pt, java_lang_Object *v) {
        int p = pt->number; /* process id */
        sq_nexts *local = (sq_nexts *) pt->data1; /* local is local to thread p */
        while (PCBfull(Q->pcb[local->nextpush,p])) {
                local->nextpush = mod(local->nextpush + 1, n)
        }
        PCBpush(v,Q->pcb[local->nextpush,p]); /* push item */
        local->nextpush = mod(local->nextpush + 1, n);
}
```

```
typedef struct {
        int in; /* counter of total number of elements inserted into PCB */
        int out; /* counter of total number of elements deleted from the PCB */
        java_lang_Object *buff[PCB_SIZE]; /* the PCB buffer, which is ``circular''*/
} PCB;
```

```
static bool_t PCBfull(PCB *b) {
        if (mod(b->in - b->out, PCB_SIZE) == PCB_SIZE - 1) /* leave empty space */
        return TRUE;
        else return FALSE;
}
```

```
static PCBpush(java_lang_Object *v, PCB *b) {
        b->buff[b->in] = v;
        b->in = mod(b-> in + 1, PCB_SIZE);
}
```

```
static int mod(int x, int n){
        while(x >= n) x = x - n;
        while(x < 0) x = x + n;
        return x;
}
```

Fig. 7

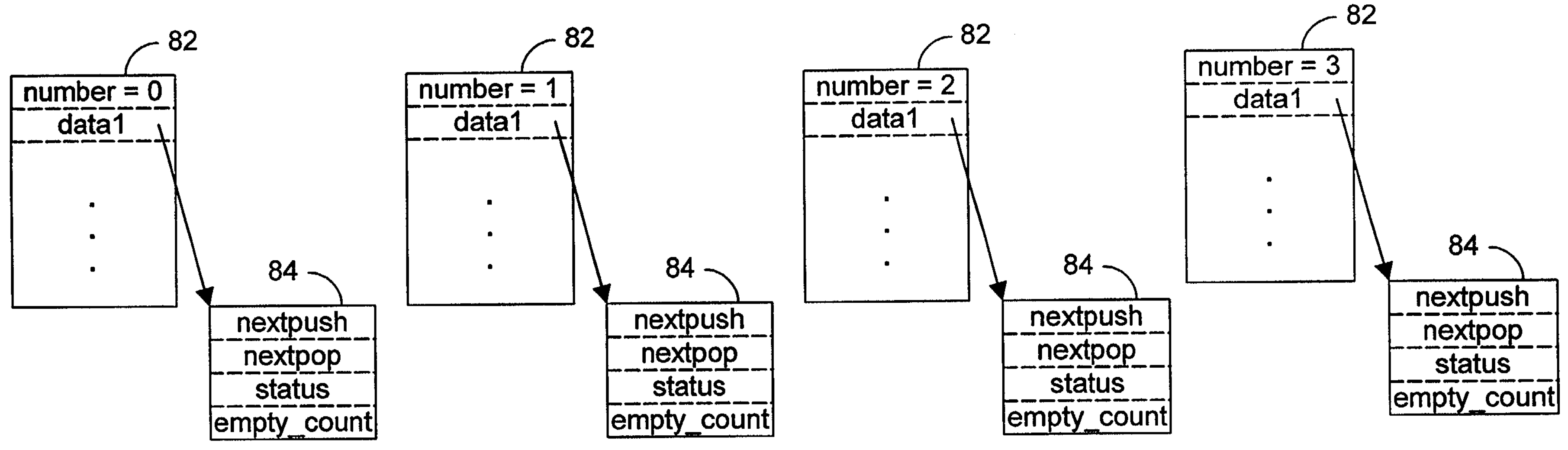

82
number = 0
data1
84
nextpush
nextpop
status
empty_count
82
number = 1
data1
84
nextpush
nextpop
status
empty_count
82
number = 2
data1
84
nextpush
nextpop
status
empty_count
82
number = 3
data1
84
nextpush
nextpop
status
empty_count

FIG. 8

```
static java_lang_Object *deque(ParallelThread *pt){
        int p = pt->number; /* process id */
        sq_nexts *local = (sq_nexts *) pt->data1; /* local is local to thread p */
        int j;
        const term_limit = 2*n; /* limit set after which termination is attempted */
        while (PCBempty(Q->pcb[p,local->nextpop])) {
                local->nextpop = mod(local->nextpop + 1,n);
                local->empty_count = local->empty_count+1;
                if (local->empty_count == term_limit) {
                        local->status = inactive;
                        for(j=0; j=n-1; j++) {
                                if (! PCBempty(Q->pcb[j,p]))
                                        local->status = active;
                        }
                        if (local->status == inactive)
                                mark_self_inactive(p,&statusBitmap);
                        if (!StatusBitmap) /* all threads are inactive, return */
                                return NULL; /* system termination state reached */
                        else local->empty_count = 0;
                }
        }
        if (local->status == inactive){
                local->status = active;
                mark_self_active(p,&statusBitmap);
        }
        int pop = local->nextpop;
        local->nextpop = mod(local->nextpop + 1, n);
        return PCBpop(Q->pcb[p,pop]);
}

static bool_t PCBempty(PCB *b) {
        return (b->in == b->out);
}

static java_lang_Object *PCBpop(PCB *b) {
        java_lang_Object *v;
        v = b->buff[b->out];
        b->out = mod(b->out +1, PCB_SIZE);
        return v;
}
```

Fig. 9

```
static void mark_self_inactive(int self, int *pStatusBitmap) {
        int oldValue,newValue;
        do {
                oldValue = *pStatusBitmap;
                newValue = oldValue & (~(1<<self));
                newValue = casInt(newValue, oldValue, pStatusBitmap);
        } while (newValue != oldValue) ;
}
```

```
static void mark_self_active(int self, int *pStatusBitmap) {
        int oldValue,newValue;
        do {
                oldValue = *pStatusBitmap;
                newValue = oldValue | (1<<self);
                newValue = casInt(newValue, oldValue, pStatusBitmap);
        } while (newValue != oldValue) ;
}
```

Fig. 10